Journal of Industrial Information Integration 26 (2022) 100253

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.elsevier.com/locate/jii

Full length article

Plant-wide interoperability and decoupled, data-driven process control with message bus communication

Petri Kannisto [a,*], David Hästbacka [a], Teresa Gutiérrez [b], Olli Suominen [c], Matti Vilkko [c], Peter Craamer [d]

[a] Faculty of Information Technology and Communication Sciences, Tampere University, P.O. Box 553, 33014, Finland
[b] Tecnalia, Basque Research and Technology Alliance (BRTA), Parque Científico y Tecnológico de Bizkaia, Astondo Bidea, 700, 48160 Derio, Spain
[c] Faculty of Engineering and Natural Sciences, Tampere University, P.O. Box 589, 33014, Finland
[d] Mondragón Sistemas, S.COOP., Ama Kandida Etorbidea 21, 20140 Andoain, Spain

ARTICLE INFO

Keywords:
Systems integration
Service-oriented architecture
Industrial cyber–physical systems
Industry 4.0
Publish–subscribe communication pattern

ABSTRACT

Conventional industrial communication systems suffer from rigidness, inflexibility and lack of scalability. The environment is heterogeneous as the systems exchange data with a variety communication protocols, some of which are proprietary. This makes it laborious and expensive to reconfigure or upgrade the systems. As the solution, this article proposes a message-bus-based communication architecture to enable information exchange between systems regardless of their geographical location and position within the functional hierarchy of the plant. The architecture not only enables communication to cross the conventional physical borders but also provides scalability to growing data volumes and network sizes. As proofs of concept, the article presents a prototype in three environments: a copper smelter, a steel plant and a distillation column. The results suggest that the message-bus-based approach has potential to renew industrial communications, a core part of the fourth industrial revolution.

1. Introduction

The industry of the future must exploit information to stay competitive and sustainable, but especially process industry suffers from non-interoperable and rigid communication systems. The lack of interoperability results from vendor-specific solutions as well as the diversity caused by the hierarchical communication structure of production plants [1]. Furthermore, the communication technologies couple systems tightly to one another, which hampers evolution [2]. The lack of loose coupling is an obstacle to the communications required for data-driven methods, which are essential for smart production [3]. In general, both data and connectivity are among the success factors of future systems [4].

To overcome the challenges, this paper studies the potential of message bus architecture in plant-wide communication. The message bus is defined as follows:

> A Message Bus is a combination of a common data model, a common command set, and a messaging infrastructure to allow different systems to communicate through a shared set of interfaces. [5]

The message bus specifies a single communication protocol and set of information models, which is conceptually simple yet powerful. This approach enforces each connected system to communicate in a unified manner. The message bus differs from Enterprise Service Bus (ESB) that implements the adapters for protocols and data in the bus itself [6]. Instead, the message bus requires a dedicated adapter for each connected system. This can require additional effort for the first integration of a system, but the cost becomes profitable once the number of connected systems grows and each can operate with the others via a single common API (Application Programming Interface).

Abbreviations: AAS, Asset Administration Shell; ACT, Advanced Control Tool; AMQP, Advanced Message Queueing Protocol; B2MML, Business to Manufacturing Markup Language; CoAP, Constrained Application Protocol; COCOP, Coordinating Optimisation of Complex Industrial Processes; ESB, Enterprise Service Bus; FMI, Functional Mockup Interface; GML, Geography Markup Language; GUI, Graphical User Interface; ICPS, Industrial Cyber–physical Systems; LCA, Life Cycle Assessment; MQTT, MQ Telemetry Transport; O&M, Observations and Measurements; RAMI 4.0, Reference Architectural Model Industrie 4.0; SOA, Service-oriented Architecture; SOAP, Simple Object Access Protocol; SOS, Sensor Observation Service; SPS, Sensor Planning Service; SWE, Sensor Web Enablement Common Data Model; TCP, Transmission Control Protocol; TSML, TimeseriesML; TSN, Time-sensitive Networking

* Corresponding author.
E-mail addresses: petri.kannisto@tuni.fi (P. Kannisto), david.hastbacka@tuni.fi (D. Hästbacka), teresa.gutierrez@tecnalia.com (T. Gutiérrez), olli.suominen@tuni.fi (O. Suominen), matti.vilkko@tuni.fi (M. Vilkko), PCraamer@msigrupo.com (P. Craamer).

https://doi.org/10.1016/j.jii.2021.100253
Received 10 July 2020; Received in revised form 1 April 2021; Accepted 26 July 2021
Available online 8 August 2021

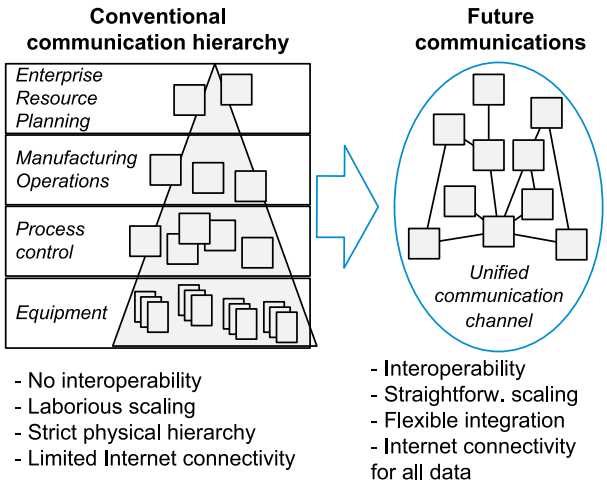

Fig. 1. A unified communication channel could bring various benefits into industrial communications.

As the result, scalability increases while the bus remains simplistic and clean from business logic.

As illustrated in Fig. 1, the goal of the message bus is to realise interoperability, eliminate physical limitations as well as enable scalable systems and Internet connectivity. This is analogous with the vision of Industrial Cyber–physical Systems (ICPS) that replace the automation pyramid with a flexible system of systems [7, p. 4]. Although the proposed message bus simplifies the overall communication structure, the approach requires the bus to manage all the complexity of data delivery. The proposed message-bus-based architecture is called COCOP (Coordinating Optimisation of Complex Industrial Processes), named after the related research project.

The research objective of this paper is:

> Design a message-bus-based communication architecture to meet the requirements and challenges of process control in plant-wide systems.

As proofs of concept, this article presents an implementation in three production systems: (1) a copper smelter, (2) a steel plant and (3) a distillation column for ethanol. (1) and (2) are actual production plants, whereas (3) operates in a research laboratory.

2. Methodology

In this study, the research method is *design science research*, which aims at designing artefacts to solve problems. Once designed, the artefacts are evaluated against their requirements to indicate usefulness. The artefacts should preferably be instantiated to demonstrate advantage. Research always aims at novelty. In design science, the novelty often comes from not the tool or method itself but rather a novel domain or fashion of application. On the other hand, design science often replies to the question if an approach or method is better than previous solutions. [8]

Design science research has three cycles that cover not only the research but also how this is connected to the environment. The relevance cycle binds the work to a tangible problem. The rigour cycle refers to connecting the work to knowledge, both by utilising existing knowledge and producing new knowledge to other practitioners. This distinguishes the work from routine design work. Finally, the design cycle is the actual design, that is, the construction of artefacts and the subsequent evaluation [9].

This study applies design science research in the following stages:

- Define the problem to be solved
  Section 3.1 Communication Needs
- Review state of art
  Section 3.2 State of Art in Systems Integration
- Design a solution (i.e., the results of the study)
  Section 4 Integration Architecture with Message Bus Communication
- Instantiate the solution (i.e., prove the results)
  Section 5 Implementations for Production Systems
- Evaluate the results
  Section 6 Discussion

3. Background

3.1. Communication needs

This section explains the requirements of plant-wide communication, including both present challenges and future directions. To have a concrete example of the challenges, a copper smelter is referred to in the text, but the issues are generalisable to industrial production systems.

3.1.1. Plant-wide coordination

In COCOP, a core goal is *plant-wide coordination* in process industry, which is challenging in certain plants. Some process plants operate multiple autonomous but interdependent unit processes. The unit processes are physically separate, and each has its own control and supervisory systems. Such plants are operable even if the control rooms communicate manually, but unexpected conditions lead to non-optimal operation if the employees lack tools to schedule production and estimate how their decisions affect the plant as a whole. Therefore, tools for plant-wide coordination help to consider both plant-wide and unit-process-specific conditions in the effort to optimise plant-wide production.

For a concrete example, let us consider the operation of a copper smelter that operates multiple unit processes (see Fig. 2). The processes are many, but to shorten the example, only *smelting furnace*, *converters* and *anode furnaces* are considered. First, smelting produces *matte* (approx. 65% copper) in a continuous process. This is delivered to converters that produce *blister copper* (approx. 99% copper) in batches. Finally, blister copper is delivered to anode furnaces, which further refine blister copper in another batch process and cast the output to anodes for subsequent electrolysis. Even the operation of a single unit process requires expertise and careful control. For instance, the converters operate in a temperature of over 1,200°C in a dirty, toxic atmosphere, which makes it non-economical to constantly take samples from the material, especially as manual samples require a break. The converters operate as batches in a cyclic fashion, and the charged materials as well as the end time of each cycle should be considered carefully, which is a difficult task to the personnel due to the lack of exact information. [10, p. 89, 127, 237]

Because our example smelter includes batch processes, *scheduling* is necessary in plant-wide coordination. Scheduling must detect the current *bottleneck* and provide sufficient time for *material handling* with cranes. Scheduling must consider the *heat balance* of the material, as too long waiting times cause solidification and too hot material causes excessive wear in the equipment. Furthermore, the *capacity* of each unit process varies unexpectedly due to defects and outages, and the *composition of raw material* varies. The scheduler should target at a suitable *composition for the output* of each unit process, as this affects how much time the subsequent unit process needs and which actions it must perform. For example, over-oxidation in converting requires reduction in the subsequent anode furnace, which is expensive. Finally, the scheduler should target at conditions that minimise the amount of pure copper that ends up to waste. The challenges of copper smelter optimisation have been studied by Korpi et al. [11].

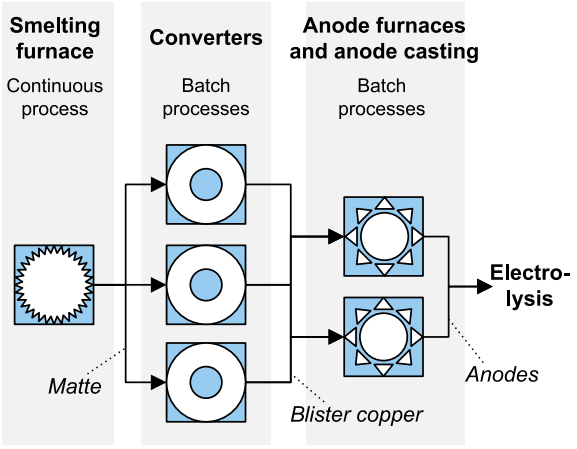

Fig. 2. A few unit processes to be coordinated in a copper smelter. Drawing based on text in [10, p. 89, 127, 237].

Plant-wide coordination requires mathematical tools for modelling and simulation that already exist, but these do not adequately support distributed optimisation in I4.0. A typical challenge is that even if it were possible to create a single mathematical tool for plant-wide optimisation, it is economically infeasible to maintain such a monolith. Furthermore, it can be desirable to distribute separate computational tasks to heterogeneous platforms depending on what platform is available in each control room or what calculation tools are applied.

Although plant-wide coordination is a special case of industrial communications, the related requirements are generalisable. Industrial systems are networked multi-vendor environments that benefit from manageable yet straightforward data exchange. Therefore, the advances that help plant-wide coordination help communications in I4.0 in general.

3.1.2. Requirements of communication

Let us look at the communication requirements of plant-wide coordination (see Fig. 3). The plant operates unit processes that are physically separate, and each has its own control and supervisory systems. The related process data is exploited in multiple locations, not only the control system but also in a scheduler system. The scheduler observes the state of all unit processes and generates schedules for the operators, helping them to consider the plant-wide scope. To develop plant operation in the future, the process data should be pushed to a cloud environment to enable process analytics and development as well as data-driven control methods. The resulting communication-related requirements are explained in the following paragraphs.

Loose coupling. As information systems are integrated, it facilitates updates and reconfiguration if the systems are connected only logically rather than physically [2]. This means that the systems do not know the physical (i.e., intrastructure-level) address of each other but communicate via a higher-level address that redirects to the physical one. In the best case however, systems should be able to communicate without knowing the communication partner in the first place. In the copper smelter example, loose coupling would help in maintaining the systems, because a system could be occasionally unavailable without requiring downtime in other systems. Downtime can affect the availability of data, which can have side effects, such as unavailability of functionality in supervisory software. Still, loose coupling helps in designing the systems to tolerate outages. Another advantage is easier testing, as any party of communication can be replaced with a mock that provides a suitable interface.

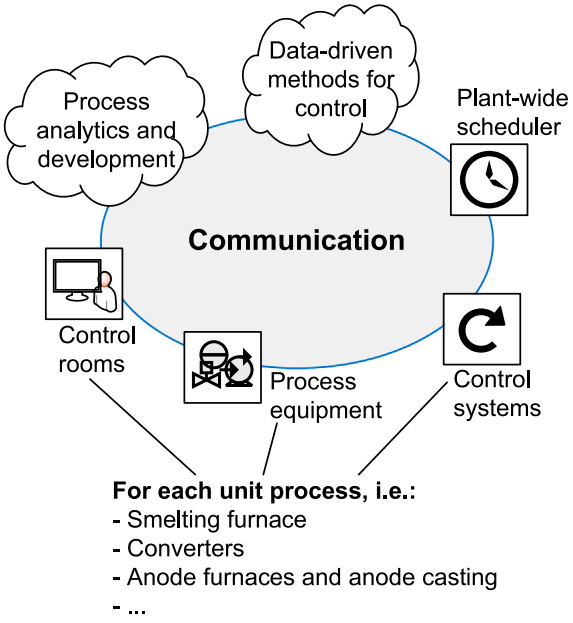

Fig. 3. The items that must communicate in the example plant.

Network scalability. Information systems tend to grow in size and data volume, which leads to a need to scale up the systems. Scaling should be straightforward, because it is not only non-economical to leave a computational reserve to each part of the system but also sometimes difficult to foresee bottlenecks. In the copper smelter example, any data source is a potential bottleneck. For instance, a server of measurement data can be overloaded by a requirement to provide measurement values to multiple clients. This would be avoided if the network infrastructure routed the measurement values to the clients, which would enable the server to concentrate on supplying the values instead of routing. The requirement of scalability becomes most evident when the data sources are computationally restricted devices, such as microcontroller-operated sensors, although all systems have a maximum capacity. It is notable, however, that the network infrastructure can become the bottleneck especially if a centralised medium takes care of message routing. Besides, such a medium can increase latency compared to a direct node-to-node communication scheme. Thus, it must be considered case by case which solution leads to the best scalability.

Internet connectivity. In the industrial context, data-driven systems lead to the exploitation of cloud infrastructures and Internet, as data is collected and processed [12]. Therefore, the networks of production systems should enable Internet connectivity. In addition, an industrial plant can span multiple factories (e.g., [11]). To coordinate such a plant, there must be a means to control the factories as a whole and deliver information between these. The example copper smelter produces anodes for electrolysis that occurs kilometres away from the plant, which necessitates coordination. On the other hand, industrial maintenance is often outsourced, which leads to a need to share equipment-related real-time data with an external organisation [13]. Finally, even supply chains necessitate collaboration [14].

Security. Security is essential in modern information technology due to the related risks. *Information security* is concerned with information, whereas *cyber security* aims at protecting the environment from threats that result from ICT [15]. For example, an exposure of information could reveal business secrets to competitors, and a malicious instance might cause a cyber incident, which could result in human, capital or environmental consequences. Therefore, the communication channel should support at least the basic security features, which are user

authentication, access control and the encryption of traffic. Whether the security measures should go beyond these depends on the case. No system can fulfil all security requirements out of the box. On the other hand, security is not only technology but also processes and education [16, pp. 22–25].

Discovery. Conventionally, systems integration has been manual in industrial plants, but once the systems become more structurally more flexible, there should be a means for the components to discover each other. In an ideal case, the components can connect to each other dynamically without any prior knowledge about the exact network address of the services they need. Discovery is most important in facilities that are reconfigured repeatedly. This need appears more often in manufacturing than in process plants, but dynamic discovery still makes reconfiguration and system updates smoother even in process industry.

Orchestration and choreography. In a distributed production plant, the systems must interact over a network. Because this functionality involves complexity, such as dependencies, preconditions, data structures and timing constraints, there should be a tool to coordinate the overall interaction. Two aspects arise: *orchestration* refers to the centralised composition of services, whereas *choreographing* helps in coordinating parties that operate autonomously but follow certain rules of interaction [17].

Standardised information models. Communications does not limit to the mechanism of sending and receiving information but must also cover presentation. The information models must cover any communication between systems. This ranges from individual measurement values to the state information of processes and the instructions of operation, such as schedules. In the copper smelter example, the schedule should covers issues, such as the timing of batch processes (i.e., converters and anode furnaces) as well as the production rate of the continuous smelting process. On the other hand, the state information of each unit process must be delivered to the scheduler, so that the generated schedules are feasible. The information models should be based on industry-wide open standards, because this enables interoperability and standards are among the key factors of Industry 4.0 [18].

Interoperability. In production plants, the typical environment is a mixture of systems from multiple vendors. For these to communicate, interoperability is necessary. This is defined as 'the ability of two or more systems or components to exchange information and to use the information that has been exchanged' [19, p. 114]. In the best case, the systems can communicate directly without any additional logic to adapt interfaces. In the copper smelter example, there are multiple control rooms and control systems. Among these, there are likely software systems from separates vendors, but information exchange is necessary to coordinate the plant. Respectively in the scope of equipment, such as valves, pumps, instrumentation and electric motors, there is a range of manufacturers. Interoperability necessitates common agreements between vendors about the practices of communication. The wider the scope of communication, the more effort is necessary to settle such agreements.

Hard real-time support. Some industrial control systems require response times that take no longer than a fraction of second. This applies particularly to devices close to the production process rather than the systems related to manufacturing operations or enterprise-level business processes. In a copper plant, the dynamics of the processes is slow due to the nature of the chemical reactions of oxidation and reduction. Therefore, the instructions of operation are often given with a resolution of minutes, but a response time should be always be guaranteed. On the other hand, certain activities require quick responses. For example, as converters oxidise matte, the generation of offgas fluctuates. Because these gases contain sulphur dioxide, they should be captured with dedicated equipment. The equipment should react appropriately to the variation of offgas generation. All in all, as the ultimate aim is to specify a communication scheme suitable for a variety of industrial plants, a hard and fast real-time support is a necessity.

3.2. State of art in systems integration

This section reviews the state of art. This includes design paradigms as well as previous frameworks and research for industrial systems integration. Although previous research successfully addresses core issues in systems integration, there is still a gap to fill in plant-wide communication systems.

3.2.1. Service-oriented architecture

Service-oriented Architecture (SOA) is a common paradigm in networked systems and suitable for industrial solutions as well. In SOA, the distributed software items are modelled as services that follow certain principles to facilitate design, maintenance and scaling. According to Erl [20], the principles are:

- Abstraction: The services should only expose to clients what is necessary to use the service
- Autonomy: Design the services to be abstract not to bind these to any specific platform, and avoid any strict runtime dependencies to other services
- Composability: Enable the services to be composed as a service built upon other services
- Discoverability: Enable the services to be discovered
- Loose coupling (or decoupling): Aim to reduce the number of direct dependencies between the services
- Reusability: Design the services generic to enable reuse
- Standardised service contracts: Use standardised technologies as the basis of service interfaces and their description
- Statelessness: Maintain state information out of the services when possible

Conventionally, SOA has been associated with the client–server model and pull communication. The technologies include, for instance, Simple Object Access Protocol (SOAP [21]), which exposes software interfaces as remotely callable procedures. SOAP is based on Hypertext Transfer Protocol (HTTP [22]) and Extensible Markup Language (XML [23]). An alternative approach is Representative State Transfer (Rest [24]), where the services are modelled as resources rather than operations. Rest, also called 'Restful' when referred with an adjective, can be based on HTTP and XML as well, but alternatives exist, such as JavaScript Object Notation (JSON [25]) for data serialisation and Constrained Application Protocol (CoAP [26]) for the communication protocol. Both SOAP and Rest, which are client–server approaches, are possible tools in designing a system that realises SOA principles.

Besides pull communication, even SOA can be implemented with the push model where the data source determines when to publish data. This suits for scenarios that require the data consumer to monitor a certain data set, such as a sensor reading. In some use cases, there must be a way to select which data to receive from a data source. This pattern is called publish–subscribe, and the protocols that implement this include, for instance, MQ Telemetry Transport (MQTT [27]) and Advanced Message Queueing Protocol (AMQP) version 0-9-1 [28].

3.2.2. Industry 4.0

Industrial Cyber–physical Systems (ICPS) have been a core trend in the research of intelligent industrial systems in recent years. ICPS, as well as the domain-agnostic term Cyber–physical Systems (CPS), refers to systems where distributed, networked, re-configurable, intelligent computational units collaborate with their environment [29]. According to Lee et al. [30], CPSs must implement connectivity and

data analysis methods to enable information collection from the environment. Information enables self-comparison within the environment, decision support and self-configuration. The implementation of ICPSs require multiple technologies, including sensors and actuators, communication protocols as well as methods for data analysis and knowledge management.

ICPSs are exploited in the German Industry 4.0 (I4.0) initiative that aims at increasing the competitiveness of industrial production. In I4.0, the technological goal is to exploit ICPSs to increase the adaptability and intelligence of production as well as facilitate timely communication within enterprises and collaborative business networks [31, pp. 13–17]. The name 'Industry 4.0' refers to the fourth industrial revolution, which does not only include technology but also organisations and management. As reviewed by Lu [32], I4.0 has multiple definitions, and some authors consider even smart cities included. The development of I4.0 is coordinated with the *Reference Architectural Model Industrie 4.0* (RAMI 4.0 [14]). RAMI 4.0 includes plant hierarchies, organisation and communication as well as the lifecycle of production equipment. In I4.0, another core concept is Asset Administration Shell (AAS), which aims to specify the interface required for information exchange in value chains [33]. AAS does not specify the interfaces required for the actual process control but instead refers to existing standards for this need.

I4.0 has competition in other countries, including Industrial Internet Consortium (IIC) in United States. To coordinate research and development, IIC has released the *Industrial Internet Reference Architecture* (IIRA [34]). IIRA overlaps partially with RAMI 4.0, but the two also complement each other as indicated in a comparative study [35]. RAMI 4.0 focuses deeply in manufacturing, whereas IIRA has a cross-domain scope without specialisation. I4.0 and IIC are not the sole initiatives to develop the industrial Internet, but this article regards them as most influential worldwide and therefore omits others. Because this work focuses on industrial production, I4.0 is the primary reference.

3.2.3. Frameworks for systems integration

To implement systems integration in I4.0, there are multiple candidate frameworks. Some of these stem from the Internet of Things (IoT) paradigm that aims at contributing to the connectivity in general whereas others focus on the domain-specific questions of industrial systems. The following paragraphs review the frameworks. Earlier, Paniagua & Delsing [36] have conducted another survey about the frameworks.

Arrowhead. Arrowhead is a SOA framework to enable the interoperability of system of systems. It provides generic services to facilitate the realisation of SOA. There are five target domains: 'production', 'smart buildings and infrastructure', 'electro mobility', 'energy production and end user services' as well as 'virtual market of energy'. To comply with Arrowhead, a system must support at least the following Arrowhead services: service registry, authorisation and orchestration. Arrowhead realises the system of systems paradigm by enabling subsystems called 'local clouds' at the side of 'global cloud'. The framework enables even legacy systems to be connected, which is important in industry. [37] Additionally, Arrowhead enables the management of workflows with choreographing and orchestration tools [38]. That is, Arrowhead provides an infrastructure with commonly needed SOA functions to enable systems to interoperate. However, it does not implement any case-specific communication, such as the delivery of production-related information in a process plant and the required communication protocol.

AUTOSAR. Automotive Open System Architecture (AUTOSAR [39]) aims to facilitate the integration of automotive software. In automotive industry, system development occurs in parallel between multiple organisations and the systems have requirements, such as safety, security, connectivity and updateability. AUTOSAR creates an abstraction layer between software and hardware to reduce dependency and promote software re-use. AUTOSAR specifies two platforms, namely classic and adaptive. The classic platform has four basic principles: 'functional safety', 'efficiency' (of development), 'field proven' (maturity) and 'performance' (e.g., hard real-time capability, scalability and support for a range of communication protocols). The adaptive platform has arisen from the need to enable the connectivity of vehicles rather than enable applications to interact within one vehicle [40]. The adaptive platform contributes to features, such as autonomous driving, automatic software updates and deployment, car-to-car communication and the 'Car-2-X' connectivity with infrastructure or smart homes.

BaSys and BaSyx. BaSyx is a reference implementation of an architecture developed in a research project called BaSys [41], which specifies a middleware to realise industrial CPSs that form a flexible network instead of the strict automation pyramid. BaSyx implements the AAS interface [33] for equipment as referred to in I4.0. The framework promises to support the communication protocols HTTP and MQTT as well as OPC UA, and it enables integration with the information systems for Manufacturing Operations Management and Enterprise Resource Planning. In addition, BaSyx provides a software development kit for developers [42]. To manage the AASs of equipment and enable discovery, BaSyx maintains a registry. The user can implement communication with 'any' protocol but must ensure compatibility between components to enable data exchange [43].

BaSyx has four levels for components. First, on the bottom, the field level covers sensors and actuators without any existing BaSys support. Second, the device level contains devices that have a BaSys-compliant interface or have been adapted to such. Third, the middleware level provides generic services, such as registry, discovery, 'protocol gateways' and AASs. Fourth, on the top, plant covers any components that 'manage, optimise and monitor' production [44].

FIWARE. FIWARE is an open-source platform that aims to facilitate the development of 'smart solutions', built by a developer community. Within the community, the explicitly mentioned domains are food production, cities, energy and industry. [45] In the core of FIWARE, there is a centralised broker component that manages context information. The context information is available for the connected components to be exploited or enriched. Furthermore, the are APIs to connect IoT and other devices. [46] The context information management API, 'Next Generation Service Interfaces Linked Data' (NGSI-LD), has even been published as a standard [47].

LWM2M. Lightweight M2M (LWM2M) has been designed to enable service architectures built on resource-constrained devices. The original motivation of the specification is the remote management of devices. LWM2M interfaces are Restful and built upon CoAP, and the specification provides a resource and information model as well. [48] Still, nowadays there is a binding for even HTTP and MQTT [49]. For data, there are serialisation formats, such as JSON and Concise Binary Object Representation (CBOR) [50]. To implement LWM2M, there are multiple open source tools that provide either a server or client [51].

OCF and IoTivity. Open Connectivity Foundation (OCF) is 'dedicated to ensuring secure interoperability for consumers, businesses and industries by delivering a standard communications platform' for IoT systems. OCF provides a set of specifications to specify aspects, such as architecture, APIs, resource model, security, discovery, adapters (i.e., 'bridging') and cloud integration. At least a subset of the specifications has been accepted as an international standard by ISO or IEC [52]. To implement communication, OCF specifies a stack that includes CoAP as the protocol and CBOR [53] as the encoding. All APIs are modelled as resources and therefore Restful. Furthermore, the resources enable the clients to subscribe for updates from the server [54].

IoTivity is an open-source reference implementation of OCF. It has been implemented in C programming language. IoTivity detaches its logic from platform-specific functionality with 'abstract interfaces' to facilitate multi-platform support and maintenance as well as lengthen the lifecycle [55]. IoTivity seems to mainly target to resource-constrai-

ned devices, as C has been chosen for the programming language. C is limited and has a non-intuitive syntax compared to modern high-level languages (such as Java, C#.NET, Python and JavaScript), but its internal logic is simple and it therefore has a wide support across platforms.

OPC UA. Open Platform Communications Unified Architecture (OPC UA) is a 'platform-independent standard through which various kinds of systems and devices can communicate' [56]. It specifies, for instance, communication protocols, information models, a security model, discovery mechanism and functionality for data browsing, aiming at interoperability in multi-vendor environments. OPC UA facilitates the realisation of interoperability with profiles that specify a set of features, so that not all compliant system must support all features. The conventional OPC UA is based on a client–server model, which is practical when single data items are explicitly requested but lacks scalability in push communication. Therefore, OPC UA PubSub [57] has been released to enable communication that scales better in data-driven applications. However, PubSub excludes message routing, which means it only standardises a data link. The original OPC UA has a production-process-oriented focus that lacks a support for the higher-level problems of production systems. This could be addressed with so called companion specifications that include an information model for devices [58], manufacturing operations [59] and robotics [60], to mention a few. Finally, the conventional OPC UA cannot guarantee real-time communication, but this can change with Time-sensitive Networking (TSN) [61]. This would enable OPC UA to implement communication down to the field level of control systems.

3.2.4. Related paradigms and research projects

Middleware is another approach to integrate software systems but suffers from maintenance-related issues. It is a system that enables software to communicate in a distributed environment [62]. In this study, middleware is considered a monolith, such as an ESB, that adapts incompatible interfaces to one another. A middleware can not only be proprietary but also implement standard interfaces, such as Web Service Information Service Bus Model (WS-ISBM [63]) that specifies an SOAP-interfaced ESB. Another Web-service-based middleware was proposed by the Socrades project [64]. On the other hand, even FIWARE can be considered a middleware, as it provides a centralised context broker [46]. While a middleware can contribute to interoperability, its maintenance can become problematic because it must implement data and protocol conversions and its hub-and-spoke approach tends to host business logic, although the medium should only deliver data [6, p. 25].

As modern industrial control necessitates the distribution of computation, multi-agent systems are a potential paradigm. They refer to systems where problems are solved by letting autonomous intelligent units called agents to collaborate. In multi-agent systems, there are challenges, such as coordination, security and task allocation [65], which are relevant in any production plant as well. Although multi-agent systems are a potential viewpoint in implementing plant-wide control systems, the paradigm itself does not remove the need to communicate information with a protocol and information model.

Prior to COCOP, earlier publications have studied industrial systems integration but lack factors considered by COCOP. The core factors are interoperability in the plant-wide scope as well as scalability to growing data volumes. The project 'Model Based Control Framework for Site-wide Optimisation of Data-intensive Processes', (MONSOON [66]) specifies a platform for the 'collection, storage and processing' of industrial big data. The system resembles a 'hub-and-spoke' architecture without an effort towards plant-wide interoperability with common technologies. The project 'Production Harmonised Reconfiguration of Flexible Robots and Machinery' (PERFoRM) proposes an industrial middleware with a common information model [67]. The proposal supports multiple communication protocols, which does not aim at reducing heterogeneity to similar extent as this study, but the common information model contributes to this goal. Theorin et al. [68] have presented 'Line Information System Architecture' (LISA). LISA uses a message bus but does not communicate with standard-based messages, and the focus is on device-level issues rather than plant-wide systems. Modoni et al. [69] propose 'Virtual Integrative Manufacturing Framework for Resources Interaction' (VICKI) to refine and deliver information between manufacturing resources. Although the framework implements messaging with a publish–subscribe-capable middleware, the focus is on production lines rather than plant-wide systems integration.

In COCOP, earlier publications have described work in progress. Initial prototypes were published in [70], comparison of candidate information models in [71] and the first specification of the architecture in [72].

4. Integration architecture with message bus communication

This section introduces the message-bus-based COCOP architecture to meet the challenges of plant-wide communications. After the architecture has been presented, the section evaluates it against the requirements explained in Section 3.1.2.

4.1. Service orientation with topic-based message bus

To implement a service-oriented plant-wide communication system, COCOP architecture has adopted the following philosophy:

> Communicate over a message bus based on standardised message structures with a standardised protocol that implements topic-based publish–subscribe communication.

The message bus has advantages over the conventional client–server-based SOA, which suffers from limitations regarding the requirements specified in Section 3.1. First, because the server must serve each client individually, it can run out of resources (e.g., [73]). The server can be scaled up, but this were more efficient if the data source remained intact and there were a scalable network medium. Second, client–server necessitates the clients to directly interact with the servers, which does not realise loose coupling. Instead, a message bus can realise SOA without similar limitations. In this approach, the message bus is the server and all other network nodes are clients regardless if they are data sources, data consumers or both. The role of the server is merely to route message traffic without any knowledge about the workflows being executed. This means that the server role disappears from the viewpoint of business logic.

Fig. 4 illustrates the message-bus-based philosophy. The message bus is in the middle, enabling communication. In the scope of COCOP, the network nodes are graphical user interfaces (GUI), optimisation software, control systems, databases and nodes that provide measurement data from equipment. Additionally, the message bus enables Internet connectivity to support geographical distribution as well as data-driven methods in the cloud.

The keywords of COCOP philosophy are *topics*, *publish–subscribe* and *standards*. Regarding topics and publish–subscribe, the message bus routes messages based on topic names. Any data sources publish messages to the topics, and the data consumers subscribe to receive the messages. Regarding standards, both the communication layer (i.e., the message bus) and the message structures are based on standards.

Fig. 5 illustrates topic-based communication within the message bus. To receive messages from one or more topics, each network node creates one or more message queues. The nodes associate their queues to the topics they subscribe for, and the message bus takes care of message routing to the queues. The message bus creates a copy of each message for each subscriber. In the figure, System A is a data source, B monitors a variable, C generates schedules based on the state of production processes and D displays the schedules created by C. The example is simplistic, as a real scenario would include too many topics and systems to be illustrated in a figure.

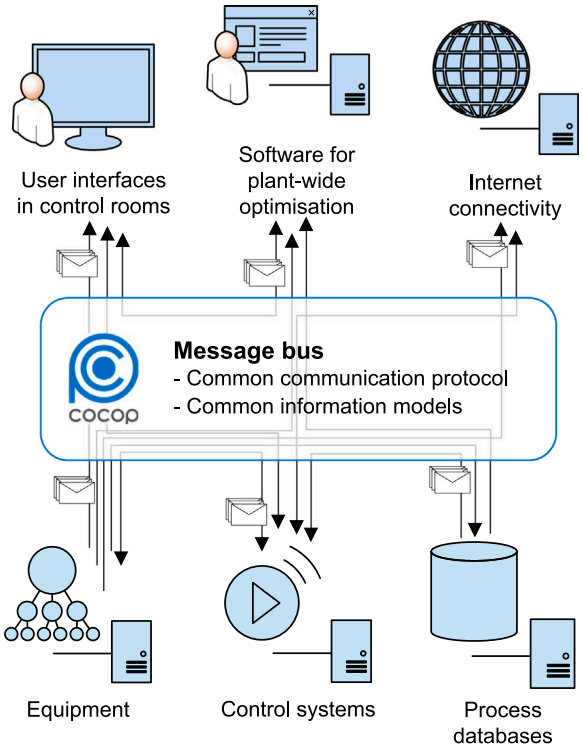

Fig. 4. COCOP architecture for plant-wide systems integration.

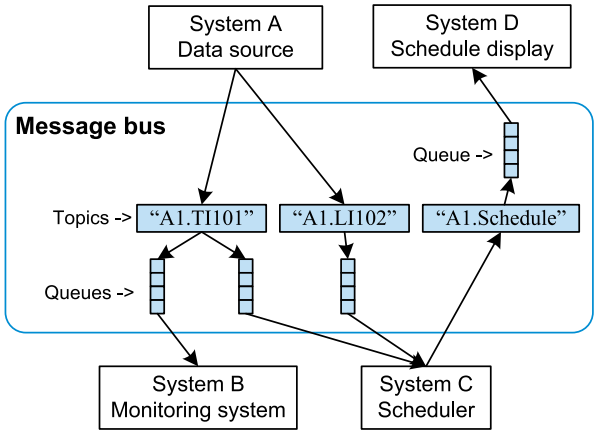

Fig. 5. An example of topic-based communication.

The novelty and advantages of COCOP stem from the goal to implement plant-wide communication with a topic-based message bus. The approach has at least three advantages. First, the network nodes are loosely coupled, and there are no direct dependencies between systems. The publish–subscribe approach enables decoupling in 'time, space and synchronisation', because the network nodes do not directly bind to one another and the queues remove any direct temporal dependencies between data sources and consumers [74]. Second, for any connected system, the message bus removes the strict levels of the automation pyramid [1]. As the third advantage, the approach scales up well for two reasons. Firstly, because there is a common data information and common protocol, all network nodes can interpret the information produced by others, which enables interoperability. This reduces the price of modifications and extensions. Secondly, even when the number of data consumers grows, there is no need to add resources to any data source, because the message bus delivers all message traffic. If the maximum capacity of the message bus is reached, it will be scaled up

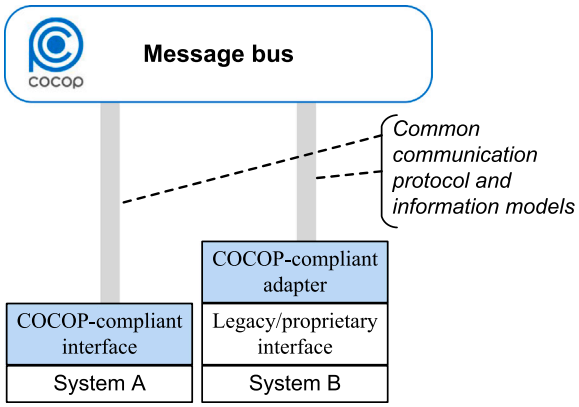

Fig. 6. If a system lacks a COCOP-compliant interface, a dedicated adapter is implemented to enable communication via the message bus. The adapters do not execute in the message bus but in another runtime.

while the data sources stay intact. Earlier, OPC UA PubSub [57] has targeted at this but lacks the specification of the actual message routing. Furthermore, COCOP is more flexible compared to ESB that tends to gather complexity by implementing protocol and data translations in itself, which leads to a rigid and difficult-to-maintain monolith.

For the communication protocol of the message bus, COCOP selects the industry standard AMQP 0-9-1, because this supports topic-based communication and has been targeted to systems that require scalability to a high volume [75]. Certainly, volume is paramount in process plants that can comprise tens or even hundreds of thousands of devices. Still, MQTT could be studied as well, because it supports the topic scheme and the newest 5.0 version (as of 2021) promises improvements in scalability [27]. To communicate with AMQP 0-9-1, RabbitMQ server [76] is a candidate for the message broker. It is available at no cost, and the source code is open.

4.2. Systems integration with adapters

A message bus enforces each connected system to communicate in a unified manner, which is different from Enterprise Service Buses (ESB) that connect systems with a comparably heavyweight monolith. Similar to a message bus, ESB performs message routing. However, an ESB can also provide services, such as a business process engine, conversion of communication protocols and mapping of message formats [6, p. 16, 28]. This makes an ESB complex compared to a message bus that contains neither business workflows nor any logic for the adaptation of data formats and protocols.

Because the message bus only uses one communication protocol and delivers messages in certain information models, adapters are necessary for some systems. This is illustrated in Fig. 6, where system A has a COCOP-compliant interface but system B does not. System B can either have a legacy interface, be vendor specific or support another standard. Although the development of adapters requires additional work, this is compensated by a lower overall complexity within the network. This is supported by a study by Trunzer et al. [77] that have applied software metrics to indicate a reduced complexity compared to point-to-point integration, which is common in the client–server scheme.

It is notable that the message bus does not execute the adapters, but these must have a separate runtime. Therefore, the internal logic of the adapters is out of the scope of this article.

4.3. Meeting the requirements

The following sections assess how the selected message bus approach meets the requirements specified in Section 3.1.2. In each section, the first paragraph assesses how COCOP meets the requirement, whereas the second paragraph compares COCOP with the frameworks reviewed in Section 3.2. The final section summarises the comparison.

Loose coupling. In conventional client–server-based web services, loose coupling is realised with domain names that map to an Internet Protocol (IP) address. In COCOP however, the systems communicate with topics and message queues that decouple the communication partners from one another. In the approach, the data sources (or publishers) associate a topic identifier to each message they publish to the message bus. Respectively, the data recipients (or subscribers) announce the message bus which topic identifiers they are interested in. To receive messages, each subscriber creates a message queue into the message bus. Based on topics, the message queue routes the messages into the queues. This publish–subscribe approach decouples systems in 'time, space and synchronisation', because the subscribers do not directly know the subscribers and they can process incoming messages from the queue as suitable for them without blocking the execution of any communication partner [74].

All the reviewed frameworks implement loose coupling to some degree, because they suggest a model where items reach one another based on a network address. For example, FIWARE is a centralised solution that bring context information available to the network. This means that any clients do not have to interact directly, but the approach is opposite to COCOP that strictly keeps the actual information storages outside of its message bus. The resource-oriented model of LWM2M and OCF implements a Restful SOA, which means that the resources can form physical one-to-one dependencies and be therefore tighter coupled than in COCOP. However, this can be relieved with a discovery mechanism, such as the one provided by Arrowhead [37], to enable the network nodes to discover one another at runtime. Arrowhead could potentially improve loose coupling even in COCOP by enabling the network nodes to discover the relevant topics at runtime. However, it requires another study to verify how feasible this combination is.

Network scalability. Scalability is one of the main arguments in COCOP. Because the bus routes all traffic based on topics, each data source just publishes data to one or more topics and any data consumers subscribe for the topics as relevant. This eliminates the need for data sources to serve each client individually. The message bus can become a bottleneck, but this will be the only node to require scaling, whereas a client–server scenario necessitates computational reserve in each data source just in case. COCOP has selected AMQP 0-9-1 as the implementing protocol, which was initially motivated by scalability and capacity [75]. If required, message bus can run on multiple servers that implement load balancing.

The message bus is scalable and can result in fraction-of-second message delivery times, but it does not scale infinitely especially to the lowest level of industrial systems. When a message travels through the bus, there is always a latency in routing, and the message must travel first into the bus and then to the ultimate recipient. This is primarily not a problem regarding scalability but temporal constraints. Furthermore, when the size of a production plant grows, a question arises how to keep topic names unique within the message bus. AMQP has mechanisms for this, but these were not yet studied for COCOP. Besides, in geographically distributed facilities, there can be separate message buses. To handle the issue of locating the message bus server, the could be a study about combining COCOP and another framework, such as Arrowhead.

Internet connectivity. COCOP message bus has been designed to support connections via Internet. Concretely, this already works with the selected AMQP interface that receives connections from clients via a protocol based on Transmission Control Protocol (TCP [78]). TCP enables connections via Internet, enabling geographically distributed systems to communicate and facilitating information exchange not only within large-scale production plants but also with external businesses that provide services. Furthermore, it is possible for sensors to stream data into the cloud, which enables data-driven methods in process control.

All of the reviewed frameworks communicate with Internet protocols. Therefore, COCOP does not bring any advantage over them but is still better than various legacy technologies, such as fieldbuses or Open Platform Communications Data Access (OPC DA), that enforce the strict levels of the automation pyramid to remain.

Security. COCOP architecture enables and supports security features but does not focus in this field. If RabbitMQ is selected as the message bus server, it enables user authentication, access control, traffic encryption and guaranteed message delivery. Additional communication-related security mechanisms can be implemented as needed. This can be based on an existing standard, such as ITU-T X.805 [79]. The organisational aspects of security are out of scope in this paper but should be implemented based on a risk assessment, which is individual to each production plant.

All of the reviewed frameworks consider security, because this is one of the enablers of architectures that connect a fleet of IoT devices with Internet-capable protocols. COCOP could be bundled with Arrowhead [37] to bring a security layer above a particular message bus, as this would bring more power to authorisation in a distributed plant. This is a topic for future research.

Discovery. Regarding discovery, COCOP is twofold. On one hand, topic-based communication enables data reception by subject without knowing the network address of the data source. On the other hand, there is no means to discover which topic names are appropriate for a particular workflow, because there is no metadata about topics.

In this aspect, COCOP is inferior compared to at least Arrowhead, BaSys and OCF. To enhance discovery, COCOP could potentially be complemented with Arrowhead or BaSys, but it requires more research to determine if this is functional in practice.

Orchestration and choreography. In COCOP, the message bus is merely a message delivery tool and cannot therefore perform anything regarding orchestration and choreography. Therefore, such actions must occur in the systems and software that interact via the message bus. On the other hand, this ensures that all business logic remains outside of the message bus, which prevents the message bus from becoming a heavyweight and difficult to maintain monolith similar to an ESB.

In this aspect, at least Arrowhead [38] is more advanced than COCOP as it provides tools for both orchestration and choreography. Therefore, there could be a future study about orchestrating and choreographing a COCOP-based system with Arrowhead.

Standardised information models. Systems integration necessitates common information models that can be serialised for to enable electronic communication. The following standards have been chosen, each enabling serialisation in XML. This article does not study information models in detail but leaves these as a subject of another article.

- *Business to Manufacturing Markup Language* for production schedules (B2MML [80]; implements ANSI/ISA-95 [1])
- *Geography Markup Language* for measurement values (GML [81])
- *Sensor Web Enablement Common Data Model* for data records (SWE [82])
- *Observations and Measurements* for metadata (O&M [83])
- *TimeseriesML* for time series (TSML [84])
- *Sensor Observation Service* for the pull communication of measurements and other data (SOS [85])
- *Sensor Planning Service* for the remote control of long-running tasks (SPS [86]).

The current standards included in COCOP provide vocabulary for the communication of industrial data, such as single measurement values, data records and timeseries, and the association of metadata, such as data quality, timestamp and location identifier. The data can be delivered either with publish–subscribe or on request. Furthermore,

B2MML specifies structures for the coordination of manufacturing operations.

Compared with other frameworks, COCOP has advantages in production-related data structures. OPC UA has a selection of information models, but the abstraction level often remains low, because OPC UA has conventionally focused on the integration of devices. OPC UA has companion specifications, such as one for ANSI/ISA-95 [59] that implements only a subset, whereas COCOP enables all structures from the B2MML serialisation format. An advantage of COCOP is its possibility to simply extend the existing vocabulary with another information model especially if this provides a schema or another comprehensive documentation, because the message bus can technically deliver any content. This can be in XML, JSON or anything else. Thus, COCOP can adopt the serialisation format of any new standard as soon as it becomes available without any additional mapping effort. COCOP brings the simplicity of a bare communication protocol while providing a selection of other features, such as message routing. However, this could degrade interoperability if any additional agreements are required regarding the usage of the new information models, especially if the information model is loose or complex. Still, because information models are many and complex, there should be another research study considering the reviewed IoT frameworks. Presumably, the frameworks focus more on general IoT questions rather than any process-industry-specific structures that are a special use case of IoT.

Interoperability. COCOP architecture realises interoperability within the production plant by enforcing all systems to communicate with a common protocol and information models. AMQP 0-9-1 is an established technology and has a wide tool support. Regarding information models, COCOP specifies a profile about the features to be supported for each standard. Still, no existing product supports COCOP, but an adapter is necessary for each, because COCOP stems from a research study without any industry-wide acceptance. However, the message bus concept can exist in an interoperable environment. Considering the advantages of COCOP, it can be asked if an existing interoperable technology, such as OPC UA, should adopt ideas from COCOP. The existing OPC UA PubSub [57] already refers to message-oriented communication but excludes the actual message bus medium and especially the management of message routing.

Of other frameworks, at least OCF and OPC UA are strong in interoperability especially if any existing products support these. Even BaSys is a potential contributor in interoperability because it implements AAS [33], which aims to realise an industrial standard for supply chains. However, AAS focuses on a higher level of communication and refers to existing standards for industrial control, including OPC UA.

Hard real-time support. COCOP does not guarantee any response times in communication due to the underlying AMQP 0-9-1 system. A guarantee would require a dedicated mechanism, although soft real time is possible, as RabbitMQ delivers messages 'immediately' as observed by a human user when not overloaded. Thus, COCOP does not fulfil the requirement of hard real-time support, and this requirement seems open as long as topic-based communication is in place, because message routing is non-deterministic. A potential solution is to bundle the message bus with a real-time capable protocol, so there would be two protocols. This is a subject of future research.

Among the reviewed frameworks, AUTOSAR supports hard real-time communication, because fast response times are necessary in certain control actions in vehicles. Furthermore, Arrowhead enables real-time communication, but this would be implemented with a technology external to Arrowhead. Although the conventional OPC UA cannot guarantee real-time communication, this is about to change due to Time-sensitive Networking (TSN) [61].

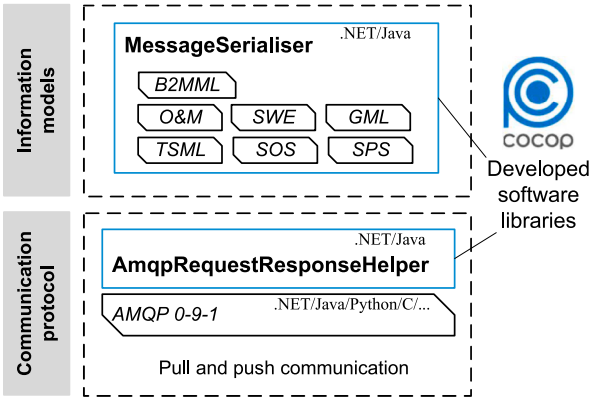

Fig. 7. COCOP communication stack consists of two layers: communication protocol and information models. These are based on existing standards.

Summary of comparison. Based on the comparison with other frameworks, COCOP could co-exist with and complement Arrowhead, BaSys, OCF and OPC UA. For COCOP, these could provide services, such as discovery, orchestration and choreography as well as authorisation. The future OPC UA is likely to provide real-time support, which is impossible with the current COCOP message bus that cannot guarantee any timing constraints. AUTOSAR supports real time but is unfavourable, because it originates from the automotive industry and therefore evokes a danger that the needs of industrial plants receive little attention in future development. LWM2M is a resource-oriented service framework. Its resource-oriented model is a direct competitor to COCOP, but this paradigm can lead to physical point-to-point dependencies, which is one of the factors that COCOP aims to avoid. FIWARE is a centralised technology and therefore different from COCOP that aims to distribute everything except message delivery. If FIWARE is considered beneficial in a production plant, it can be integrated with COCOP, but generally there is no reason to combine these two. Compared to other technologies, COCOP excels at loose coupling, the scalability of message routing and the flexibility of adding new information models.

5. Implementations for production systems

COCOP has delivered multiple implementations for everyday industrial use cases. The enabler of implementations is a communication stack for plant-wide communications. The actual proofs of concept were developed for the coordination of a copper smelter, quality control in a steel plant and Online Life Cycle Assessment related to a distillation column.

5.1. Open communication stack and toolkit

To maintain interoperability and facilitate software development, a communication stack was developed to cover both the communication protocol and information models (Fig. 7). The communication protocol layer is based on the client libraries made for AMQP 0-9-1 or particularly RabbitMQ. These libraries exist for multiple environments, including .NET, Java, C++, C and Python, to mention a few. In case pull communication is needed, the developers can utilise the *AmqpRequestResponseHelper* library implemented in COCOP for C#.NET and Java. Furthermore, COCOP delivers a library called *MessageSerialiser* to process the messages exchanged between application. This enables the developers to concentrate on the application logic rather than the message syntax. MessageSerialiser was implemented in C#.NET and Java, and it implements information models based on the standards presented in Section 4.3: B2MML, O&M, SWE, GML, TSML, SOS and SPS.

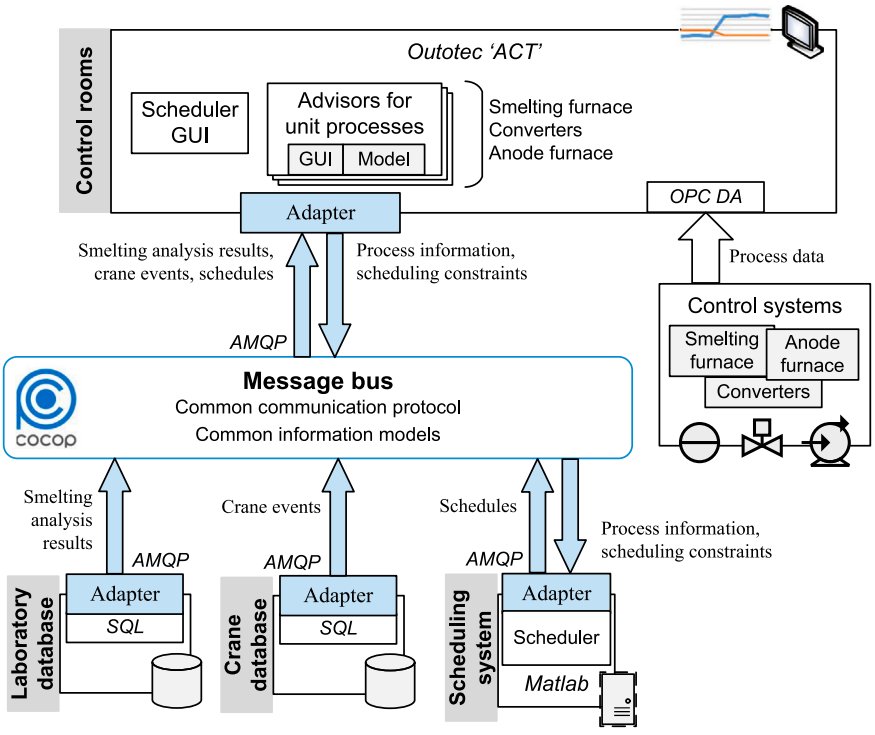

Fig. 8. The software tools for the coordinating optimisation of a copper smelter.

The COCOP-implemented parts of the stack have been published as open source.[1] as a part of *COCOP Toolkit* [87] Thus, the software is available for other studies and further development as needed. Additionally, COCOP Toolkit includes not only the stack but also other tools, such as a message logger application and a library to connect Matlab with the message bus protocol.

5.2. Plant-wide optimisation in copper smelter

This prototype aims at the plant-wide coordination of a copper smelter, providing advisory to control the unit processes smelting furnace, converters as well as anode furnace and anode casting. As explained in Section 3.1, operating a copper smelter requires professional knowledge and timely reactions to the repeatedly changing conditions. The current control tools are far from optimal, but more advanced functionality can be implemented with mathematical models and algorithms.

To help operators, there must be control tools that exploit process-related measurement data and react appropriately to events within the plant. The process data is available in the control system of each unit process. In addition, there is a crane database that reveals how material moves and another for the composition of the matte produced by smelting furnace. Besides the operation of unit processes, there must be a scheduler to coordinate the plant as a whole. Schedules are essential because converters and anode furnaces operate in batches rather than continuously. The timing of each batch must cover factors, such as the availability of the input, the prevailing capacity of the subsequent unit process and the heat balance of the material. Therefore, the schedules must consider not only the actual process states but also future performance regarding factors, such as productivity, copper losses and energy consumption.

Fig. 8 illustrates the tools developed as well as how these are connected with the message bus. The advisory tools and the scheduler GUI execute in a platform called Outotec 'ACT' (Advanced Control Tool). The advisory tools receive process data via the legacy OPC DA protocol (in the future, the plan is to replace this link with the message bus). All other data travels via the message bus, including analysis results from the smelting furnace, crane events and schedules. Furthermore, the scheduler receives the state information of unit processes from ACT. In addition, the operators feed any operation-related constraints, such as maintenance breaks, with the scheduler GUI that publishes these to the message bus and become available to the scheduler.

To communicate via the message bus, an adapter was created for each connected system. The adapters provide message bus connectivity for Structured Query Language (SQL) interfaces, a Matlab mathematics environment and the proprietary ACT interface. All messages are processed with the *MessageSerialiser* library of COCOP Toolkit, Matlab with the Java implementation and others with .NET. This realises interoperability within the multi-platform and multi-vendor environment.

Once the system had been developed, a two-week online test period was organised in the copper smelter. Two weeks is short in a copper smelter due to the complexity of production processes, so a longer test could be committed to prove long-term improvements. However, the advisory tools as well as the scheduler reacted to plant-wide states and events appropriately most of the time. Due to the complexity of the processes, there is still room for improvement in the mathematical tools, but the experiment proved the architecture concept to be functional. Furthermore, there are advantages in interoperability and loose coupling, as there are no direct dependencies between the systems connected via the message bus.

5.3. Quality control in steel plant

The steel plant prototype aims to reduce the number of surface defects in micro-alloyed steel products. The prototype improves the performance of the three sub-processes that affect the generation of defects: *secondary metallurgy*, *continuous casting* and *hot rolling*. To achieve this objective, several predictive models were developed. Fig. 9 shows the final three-layer architecture, where each layer operates in a separate virtual machine:

[1] https://kannisto.github.io/Cocop-Toolkit/.

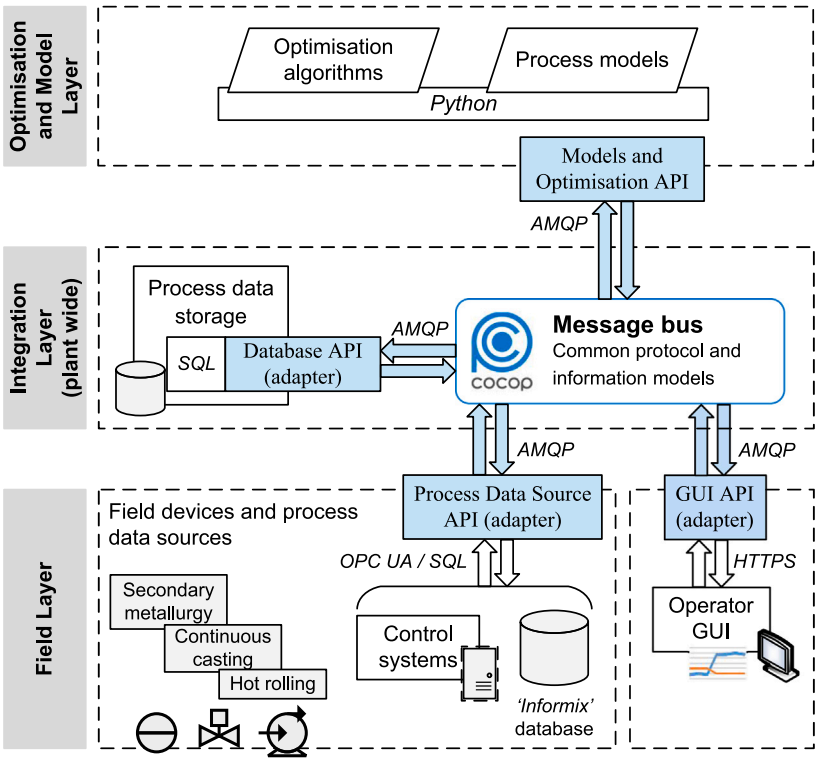

Fig. 9. The software tools for the quality control of a steel plant.

1. **Field Layer** represents the operations near the production line, covering data acquisition and a user interface for the operators.
2. **Integration Layer** integrates information from the various data sources and stores this in a centralised data container called Process Data Storage. All systems integration occurs via the message bus.
3. **Optimisation and Model Layer** is where the optimisation takes place. The outcome is stored in Process Data Storage and shown in the operator GUI.

To enable systems integration via the message bus, adapters were developed (also called Application Programming Interfaces or APIs). First, the operator GUI communicates via web services based on Hypertext Transfer Protocol Secure (HTTPS). Second, control systems provide process data over OPC UA. Third, the Informix database has an SQL interface.

There are two modes of operation:

1. **Online Mode** (for online tools): The operator uses the user interface to request to start calculation. Then, the user interface continuously receives results calculated from actual process data.
2. **Offline Mode** (for optimisation and offline tools): The operator changes one or more variables in the user interface and launches the calculation of the related model or optimiser that returns a result.

For both operation modes, a message structure was defined, and the required message queues were created in the message bus. The messages are routed based on topic names. The prototype does not exploit the data structures of COCOP MessageSerialiser library but still communicates over the common protocol.

To test the tools, an online test period was organised in a steel plant. Not only the tools and communication architecture were proven functional but also an indication was discovered that the generated advisory reduces the amount of re-working and reject as intended.

5.4. Online life cycle assessment

Online Life Cycle Assessment (online LCA) enables the active monitoring of the environmental impacts of production. The conventional 'offline' LCA refers to the estimation of the environmental impacts caused by an entity during its entire lifecycle. While useful, this approach does not actively guide plant operators due to the pre-calculated nature. In contrast, online LCA estimates the impact during operation based on the actual state of the production process. This enables the operators to monitor the effects of their decisions. Such monitoring is a suitable application for COCOP message bus.

This prototype performs online LCA related to a distillation column that extracts ethanol from ethanol–water mixture. The column is located in a laboratory but operates actual industrial equipment, such as pumps and valves as well as a Valmet 'DNA' control system. In the prototype, LCA is based on the values of energy consumption and raw material feed.

The message bus connects three components: the production process, an LCA server and a control room (see Fig. 10). The production process provides measurement values that are delivered to the LCA server to enable LCA and Outotec ACT. ACT visualises both the measurement values and the LCA results, so that the operators can see how these are related. To enable connectivity via the message bus, there is an adapter for each component. The adapter of the production process accesses measurements via OPC UA, whereas the LCA model is controlled over Functional Mockup Interface (FMI), which is a specification for exchanging simulation-related data. The LCA model was modelled in a tool called Sulca and exported to enable execution as a standalone application. The message bus performs message routing based on topics, which decouples the data producers and consumers.

A practical experiment was performed with the prototype. The column was heated to distil ethanol from water–ethanol mixture. As the heating power of the column was changed, clear step responses occurred in the calculated LCA results. This experiment demonstrates the capability of the message bus for geographical distribution, as the production process was located 200 kilometres away from the LCA

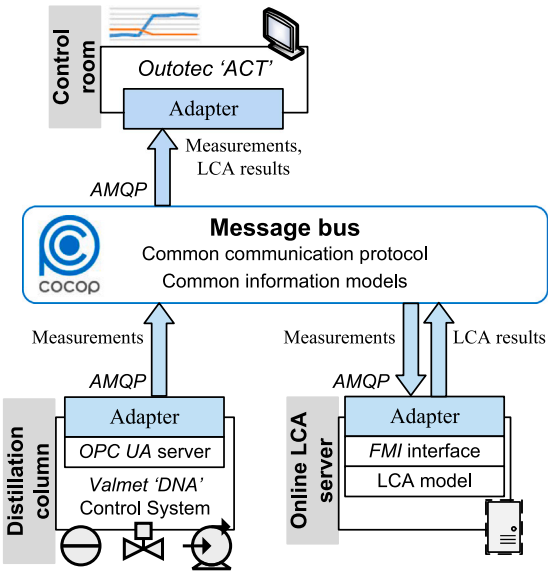

Fig. 10. The online LCA prototype.
Source: Figure modified from [88].

server and the control room where monitoring occurred. On the other hand, the connected systems are actual products and equipment for industrial process control, which shows that the message bus suits for a production environment. More importantly, the prototype shows how heterogeneous environments can be connected in a loosely coupled manner.

6. Discussion

The research objective of this article is:

Design a message-bus-based communication architecture to meet the requirements and challenges of process control in plant-wide systems.

As a solution to the requirements of plant-wide systems presented in Section 3.1, Section 4 introduced the message-bus-based COCOP architecture. The architecture follows the SOA principles that have been proven beneficial in the integration of enterprise systems [20]. The message bus applies topics to perform all of the required message routing between software systems. This ensures that the systems can reach one another with a single standardised communication protocol. Furthermore, the systems communicate with common information models, although it is straightforward to extend the architecture with new message structures as required. Therefore, the message bus simplifies the topology of communication and ensures interoperability within the production plant. This is an advantage compared to conventional point-to-point connections or the hub-and-spoke approach, which performs data and protocol adaptions in a centralised node typical to ESBs.

To prove the architecture, this paper presented three prototypes as well as a software toolkit in Section 5. The prototypes include the plant-wide optimisation of a copper smelter, quality control in a steel plant and online LCA. In each prototype, multiple software systems were connected via the COCOP message bus, providing a single medium to enable interoperability. It was necessary to develop interface adapters, but this burden was compensated as the systems became reachable within the entire plant despite their legacy interface. On the other hand, the software tools of *COCOP Toolkit* reduced the development effort required for the implementation of the adapters. The experiments did not test the capacity of the message bus, because the message volumes were at most tens per minute. Therefore, there could be another study about the performance of the message bus in plant-wide communications with tens of thousands of network nodes that stream data frequently.

The concept of COCOP excels particularly at loose coupling, scalability and interoperability. Together with Internet connectivity, these factors facilitate the development of data- and event-driven solutions, which are gaining importance in industrial production [12,68]. COCOP helps in the development of plant-wide systems where the functions must span units, sites and Internet. This opens possibilities to the more extensive use of cloud computing, advanced analytics and machine learning applications that require massive amounts of data. The problems COCOP aims to solve resemble the goals of OPC UA PubSub [57]. PubSub relaxes the conventional strict client–server model of OPC UA but lacks the exact specification of message routing. Conversely, COCOP exclusively specifies topic-based messaging. Another advantage of COCOP is the ease of adding new information models especially when the information model provides a concrete schema or another specification for serialisation. In OPC UA, this requires an explicit mapping to the OPC UA information model, such as the one published for ANSI/ISA-95 [59]. However, to apply an information model in an interoperable manner, even COCOP may necessitate a dedicated profile to explicitly specify the utilisation, especially if the model is either loose or too complex to be specified with a schema language.

COCOP could be extended with other frameworks as it lacks a full support for the requirements of plant-wide systems (see Section 4.3). COCOP focuses on concrete information exchange rather than higher-level SOA features, such as orchestration, choreography and discovery, and the current COCOP does not enable hard real time communication either. Therefore, COCOP could include another framework to provide the missing features rather than developing yet another solution. Earlier, there have been studies about combining diverse frameworks and communication protocols, such as a publication about combining Arrowhead and OPC UA [89]. Based on the comparison in Section 4.3, the potential extenders of COCOP include, for instance, Arrowhead for discovery, orchestration and authorisation, OPC UA with TSN for real-time support and BaSys to implement Asset Administration Shell [33] for interoperability within supply chains. These potential extensions require future research for verification.

COCOP contributes to the I4.0 aspects that are related to communications and the integration of physical systems across the hierarchical levels of production systems. Fig. 11 illustrates this within the reference architecture RAMI 4.0, which specifies hierarchy levels from shop floor to production coordination and collaboration, layers from physical assets to business and the life cycle of both products and physical entities [14, pp. 5–11]. In layers, *asset* covers physical assets and people and *business* focuses on business processes, so these are excluded in COCOP. COCOP resides in other layers from *communication* to *functional* by enabling the vertical integration of systems with a communication protocol and information models. In hierarchy levels, COCOP covers everything from product to enterprise, as the intention is to provide a single solution for all integration within the plant. In the future, it could be studied how COCOP could support business-to-business information exchange and how the architecture should be extended in this case. In addition, COCOP has not yet been implemented in the device level. Regarding life cycle, COCOP operates in the *usage* of systems, as neither the design, maintenance nor manufacturing of production tools is considered, but these could be supported with additional information models. In summary, COCOP aims at solving the issues of vertical and horizontal integration of systems, which have constantly been problematic in production systems.

7. Conclusions

This article presented the message-bus-based COCOP architecture for plant-wide communications. The architecture enables the integration of industrial ICT systems over the levels of the conventional

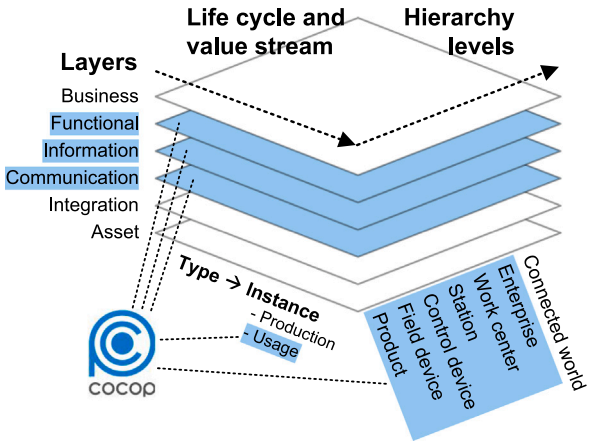

Fig. 11. In RAMI 4.0, COCOP architecture contributes to the layers communication, information and functional across all hierarchy levels within an enterprise in the usage of facilities.
Source: Re-drawn and modified from [90, p. 43].

automation hierarchy as well as geographical borders. The core of the architecture is a message bus that provides a single medium for all communication. This helps in the management of heterogeneity, enabling a loosely coupled, adaptable, interoperable environment. As proofs of concept, COCOP was applied in three use cases in process industry.

There are still topics to research in the future. The proposed architecture does not implement all the requirements of plant-wide SOA for Industry 4.0, but it could be complemented with another integration framework, such as Arrowhead, BaSys or OPC UA. This could enhance features, such as discovery, orchestration and authorisation. To widen the applicability of COCOP, there could be a study to explore existing standardised information models. To bring COCOP to the shop floor, there could be a study about connecting devices with limited computational resources and enable communication with hard real-time requirements. Additionally, there could be proofs of concept in manufacturing, as the current use cases only demonstrate applicability in process industry. Finally, there could be experiments that involve data-driven control methods, which will gain importance in the future [3].

CRediT authorship contribution statement

Petri Kannisto: Conceptualization, Writing – original draft, Writing – review & editing, Software. **David Hästbacka:** Conceptualization, Writing – review & editing, Supervision. **Teresa Gutiérrez:** Writing – original draft. **Olli Suominen:** Software. **Matti Vilkko:** Project administration, Supervision. **Peter Craamer:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723661. This study reflects only the authors' views, and the Commission is not responsible for any use that may be made of the information contained therein. The authors want to express their sincere gratitude to the funder and the project partners in COCOP project (Coordinating Optimisation of Complex Industrial Processes, https://www.cocop-spire.eu/).

Additionally, this work was supported by the Academy of Finland (grant 310098).

Finally, multiple persons participated in the development of the prototypes. The authors would like to mention especially the following: Mikko Korpi, Jussi Leinonen and Ari Kruskopf from Metso Outotec, Finland; Felix Bayon and Asier Arteaga from Sidenor Aceros Especiales S.L., Spain; Ville Mörsky from Tampere University; Kari Rainio, Matias Alarotu, Tiina Pajula and Jouni Savolainen from VTT Technical Research Centre of Finland.

References

[1] ANSI/ISA-95.00.01, Enterprise-control system integration: Part 1: Models and terminology, International Society of Automation, 2010.
[2] H. Panetto, B. Iung, D. Ivanov, G. Weichhart, X. Wang, Challenges for the cyber-physical manufacturing enterprises of the future, Annu. Rev. Control 47 (2019) 200–213, http://dx.doi.org/10.1016/j.arcontrol.2019.02.002.
[3] Y. Cheng, K. Chen, H. Sun, Y. Zhang, F. Tao, Data and knowledge mining with big data towards smart production, J. Ind. Inf. Integr. 9 (2018) 1–13, http://dx.doi.org/10.1016/j.jii.2017.08.001.
[4] D. Gürdür Broo, U. Boman, M. Törngren, Cyber-physical systems research and education in 2030: Scenarios and strategies, J. Ind. Inf. Integr. 21 (2021) 100192, http://dx.doi.org/10.1016/j.jii.2020.100192.
[5] G. Hohpe, B. Woolf, Enterprise integration patterns - message bus, 2019, https://www.enterpriseintegrationpatterns.com/patterns/messaging/MessageBus.html [Retrieved 8 Jul 2020].
[6] A. Boyd, D. Noller, P. Peters, D. Salkeld, T. Thomasma, C. Gifford, S. Pike, A. Smith, SOA in Manufacturing Guidebook, 2008, ftp://public.dhe.ibm.com/software/plm/pdif/MESA_SOAinManufacturingGuidebook.pdf [Retrieved 11 May 2020].
[7] Cyber-physical Systems, Chancen und Nutzen aus Sicht der Automation, VDI Verein Deutscher Ingenieure e.V., 2017, https://www.vdi.de/ueber-uns/presse/publikationen/details/cyber-physical-systems-chancen-und-nutzen-aus-sicht-der-automation [Retrieved 30 Oct 2020].
[8] S.T. March, G.F. Smith, Design and natural science research on information technology, Decis. Support. Syst. 15 (4) (1995) 251–266, http://dx.doi.org/10.1016/0167-9236(94)00041-2.
[9] A.R. Hevner, A three cycle view of design science research, Scand. J. Inform. Syst. 19 (2) (2007) 87–92.
[10] M.E. Schlesinger, M.J. King, K.C. Sole, W.G. Davenport, Extractive Metallurgy of Copper, fifth ed., Elsevier, Oxford, 2011, http://dx.doi.org/10.1016/B978-0-08-096789-9.10008-3.
[11] M. Korpi, O. Suominen, J. Jansson, J. Pihlasalo, M. Vilkko, Plant-wide optimization of a copper smelter: How to do it in practice? Proceedings, European Metallurgical Conference EMC 2019 Vol. 1, 2019, pp. 95–106.
[12] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, J. Manuf. Syst. 48 (2018) 157–169, http://dx.doi.org/10.1016/j.jmsy.2018.01.006.
[13] P. Kannisto, D. Hästbacka, A. Marttinen, Information exchange architecture for collaborative industrial ecosystem, Inform. Syst. Front. 22 (3) (2020) 655–670, http://dx.doi.org/10.1007/s10796-018-9877-0.
[14] P. Adolphs, H. Bedenbender, D. Dirzus, M. Ehlich, U. Epple, M. Hankel, R. Heidel, M. Hoffmeister, H. Huhle, B. Kärcher, H. Koziolek, R. Pichler, S. Pollmeier, F. Schewe, A. Walter, B. Waser, M. Wollschlaeger, Reference architecture model industrie 4.0 (RAMI4.0), VDI Verein Deutscher Ingenieure e.V./ZVEI – German Electrical and Electronic Manufacturers' Association, 2015, https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/GMA_Status_Report_Reference_Archtitecture_Model_Industrie_4.0_RAMI_4.0_/GMA-Status-Report-RAMI-40-July-2015.pdf, [Retrieved 18 May 2020].
[15] R. von Solms, J. van Niekerk, From information security to cyber security, Comput. Secur. 38 (2013) 97–102, http://dx.doi.org/10.1016/j.cose.2013.04.004.
[16] NSTISSI no. 4011, National Training Standard for Information Systems Security (INFOSEC) Professionals, National Security Telecommunications and Information Systems Security Committee, Fort George G. Meade, MD, USA, 1994.
[17] R. Dijkman, M. Dumas, Service-oriented design: a multi-viewpoint approach, Int. J. Coop. Inf. Syst. 13 (04) (2004) 337–368, http://dx.doi.org/10.1142/S0218843004001012.

[18] A.A. Nazarenko, J. Sarraipa, L.M. Camarinha-Matos, C. Grunewald, M. Dorchain, R. Jardim-Goncalves, Analysis of relevant standards for industrial systems to support zero defects manufacturing process, J. Ind. Inf. Integr. 23 (2021) 100214, http://dx.doi.org/10.1016/j.jii.2021.100214.

[19] IEEE Standard Computer Dictionary IEEE Std 610, A Compilation of IEEE Standard Computer Glossaries, The Institute of Electrical and Electronics Engineers, 1991, http://dx.doi.org/10.1109/IEEESTD.1991.106963.

[20] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, Upper Saddle River, NJ, 2005.

[21] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, W3C, 2000, https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ [Retrieved 18 May 2020].

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, RFC 2616: Hypertext Transfer Protocol – HTTP/1.1, The Internet Society, 1999, https://tools.ietf.org/html/rfc2616 [Retrieved 18 May 2020].

[23] T. Bray, J. Paoli, C. Sperberg-McQueen, Y. Mailer, F. Yergeau (Eds.), Extensible markup language (XML) 1.0, 5th edition, W3C, 2008, https://www.w3.org/TR/2008/REC-xml-20081126/ [Retrieved 18 May 2020].

[24] R. Fielding, Architectural Styles and the Design of Network-Based Software Architectures (Ph.D. thesis), University of California, Irvine, 2000.

[25] RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format, Internet Engineering Task Force (IETF), 2017, https://tools.ietf.org/html/rfc8259 [Retrieved 18 May 2020].

[26] Z. Shelby, K. Hartke, C. Bormann, RFC 7252: The Constrained Application Protocol (CoAP), Internet Engineering Task Force (IETF), 2014, https://tools.ietf.org/html/rfc7252 [Retrieved 18 May 2020].

[27] MQTT version 5.0, OASIS Standard, 2019, https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html [Retrieved 27 May 2020].

[28] AMQP, Advanced Message Queueing Protocol Version 0-9-1, AMQP Working Group 0-9-1, 2008, http://www.amqp.org/specification/0-9-1/amqp-org-download [Retrieved 27 May 2020].

[29] L. Monostori, Cyber-physical production systems: Roots, expectations and R&D challenges, Procedia CIRP 17 (2014) 9–13, http://dx.doi.org/10.1016/j.procir.2014.03.115.

[30] J. Lee, B. Bagheri, H.-A. Kao, A cyber-physical systems architecture for industry 4.0-based manufacturing systems, Manuf. Lett. 3 (2015) 18–23, http://dx.doi.org/10.1016/j.mfglet.2014.12.001.

[31] Recommendations for implementing the strategic initiative INDUSTRIE 4.0, Final Report of the Industrie 4.0 Working Group, Acatech – National Academy of Science and Engineering, 2013, https://www.acatech.de/wp-content/uploads/2018/03/Final_report_Industrie_4.0_accessible-1.pdf [Retrieved 15 May 2020].

[32] Y. Lu, Industry 4.0: A survey on technologies, applications and open research issues, J. Ind. Inf. Integr. 6 (2017) 1–10, http://dx.doi.org/10.1016/j.jii.2017.04.005.

[33] Details of the Asset Administration Shell, Part 1 – The Exchange of Information Between Partners in the Value Chain of Industrie 4.0 (Version 2.0), Federal Ministry for Economic Affairs and Energy (BMWi), 2019, https://www.zvei.org/en/press-media/publications/details-of-the-asset-administration-shell/ [Retrieved 11 Jan 2021].

[34] S.-W. Lin, B. Miller, J. Durand, G. Bleakley, A. Chigani, R. Martin, B. Murphy, M. Crawford, The Industrial Internet of Things Volume G1: Reference Architecture, Industrial Internet Consortium, 2017, https://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf [Retrieved 19 May 2020].

[35] S.-W. Lin, B. Murphy, E. Clauer, U. Loewen, R. Neubert, G. Bachmann, M. Pai, M. Hankel, Architecture Alignment and Interoperability: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper, Industrial Internet Consortium, 2017, https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf [Retrieved 19 May 2020].

[36] C. Paniagua, J. Delsing, Industrial frameworks for internet of things: A survey, IEEE Syst. J. 15 (1) (2021) 1149–1159, http://dx.doi.org/10.1109/JSYST.2020.2993323.

[37] P. Varga, F. Blomstedt, L.L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, I.M. de Soria, Making system of systems interoperable – the core components of the arrowhead framework, J. Netw. Comput. Appl. 81 (2017) 85–95, http://dx.doi.org/10.1016/j.jnca.2016.08.028.

[38] D. Kozma, P. Varga, K. Szabó, Achieving flexible digital production with the arrowhead workflow choreographer, in: IECON 2020 the 46th Annual Conference of the IEEE Industrial Electronics Society, 2020, pp. 4503–4510, http://dx.doi.org/10.1109/IECON43393.2020.9254404.

[39] AUTOSAR Introduction, The Vision, the Partnership and Current Features in a Nutshell, 2020, https://www.autosar.org/fileadmin/ABOUT/AUTOSAR_EXP_Introduction102020.pdf [Retrieved 18 Mar 2021].

[40] S. Fürst, M. Bechter, AUTOSAR for connected and autonomous vehicles: The AUTOSAR adaptive platform, in: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), 2016, pp. 215–217, http://dx.doi.org/10.1109/DSN-W.2016.24.

[41] BaSys 4.0, Basissystem industrie 4.0, 2021, https://www.basys40.de/ [Retrieved 18 Mar 2021].

[42] BaSyx, Overview, 2021, https://wiki.eclipse.org/BaSyx_/_Overview [Retrieved 18 Mar 2021].

[43] BaSyx, Concepts, 2020, https://wiki.eclipse.org/BaSyx_/_Concepts [Retrieved 18 Mar 2021].

[44] BaSyx, What is BaSyx, 2020, https://wiki.eclipse.org/BaSyx_/_WhatIsBasyx [Retrieved 19 Mar 2021].

[45] FIWARE, About us, 2021, https://www.fiware.org/about-us/ [Retrieved 19 Mar 2021].

[46] FIWARE, Developers, 2021, https://www.fiware.org/developers/ [Retrieved 19 Mar 2021].

[47] ETSI GS CIM 009, Context Information Management (CIM); NGSI-LD API, V1.1.1, European Telecommunications Standards Institute, 2019.

[48] Lightweight M2M, LWM2M, 2021, https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/ [Retrieved 19 Mar 2021].

[49] OMA-TS-LightweightM2M_Transport-V1_2-20201110-A, Lightweight Machine to Machine Technical Specification: Transport Bindings, Approved Version: 1.2, Open Mobile Alliance, 2020, http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Transport-V1_2-20201110-A.pdf [Retrieved 19 Mar 2021].

[50] OMA-TS-LightweightM2M_Core-V1_2-20201110-A, Lightweight Machine to Machine Technical Specification: Core, Approved Version: 1.2, Open Mobile Alliance, 2020, http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf [Retrieved 19 Mar 2021].

[51] LwM2M, Open source code, 2021, https://github.com/OpenMobileAlliance/OMA_LwM2M_for_Developers/wiki/LwM2M-Open-Source-Code [Retrieved 19 Mar 2021].

[52] OCF, Open connectivity foundation, 2021, https://openconnectivity.org/ [Retrieved 19 Mar 2021].

[53] C. Bormann, P. Hoffman, RFC 7049: Concise Binary Object Representation (CBOR), The Internet Society, 2013, https://tools.ietf.org/html/rfc7049 [Retrieved 19 Mar 2021].

[54] OCF Specification Overview, Core technology specification, 2019, https://openconnectivity.org/wp-content/uploads/2019/03/2.-OCF-Architecture-Introduction.pdf [Retrieved 19 Mar 2021].

[55] IoTivity, Architecture, 2021, https://iotivity.org/about/iotivity-architecture [Retrieved 19 Mar 2021].

[56] OPC UA, OPC Unified Architecture Specification Part 1, Overview and Concepts, Release 1.04, OPC Foundation, 2017.

[57] OPC UA Pt.14, OPC unified architecture specification part 14, PubSub, Release 1.04, OPC Foundation, 2018.

[58] OPC unified architecture Part 100, Devices, Release 1.02, OPC Foundation, 2019.

[59] OPC unified architecture for ISA-95, Common object model specification, Release 1.00, OPC Foundation, 2013.

[60] OPC UA for robotics part 1, Vertical integration, Release 1.00, OPC Foundation, 2019.

[61] D. Bruckner, M. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, T. Sauter, An introduction to OPC UA TSN for industrial communication systems, Proc. IEEE 107 (6) (2019) 1121–1131, http://dx.doi.org/10.1109/JPROC.2018.2888703.

[62] G. Hohpe, B. Woolf, Enterprise integration patterns - introduction, 2019, https://www.enterpriseintegrationpatterns.com/patterns/messaging/Introduction.html [Retrieved 29 Mar 2021].

[63] WS-ISBM 1.0, Web Service Information Service Bus Model, OpenO&M Standard, 2014, http://www.openoandm.org/files/standards/ISBM-1.0.pdf [Retrieved 29 Mar 2021].

[64] L.M.S. de Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, D. Savio, SOCRADES: A web service based shop floor integration infrastructure, in: The Internet of Things: First International Conference, IOT 2008, Zurich, Switzerland, Proceedings, Springer Berlin Heidelberg, 2008, pp. 50–67, http://dx.doi.org/10.1007/978-3-540-78731-0_4.

[65] A. Dorri, S.S. Kanhere, R. Jurdak, Multi-agent systems: A survey, IEEE Access 6 (2018) 28573–28593, http://dx.doi.org/10.1109/ACCESS.2018.2831228.

[66] M. Sarnovsky, P. Bednar, M. Smatana, Big data processing and analytics platform architecture for process industry factories, Big Data Cogn. Comput. 2 (1) (2018) 3, http://dx.doi.org/10.3390/bdcc2010003.

[67] F. Gosewehr, J. Wermann, W. Borsych, A.W. Colombo, Specification and design of an industrial manufacturing middleware, in: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), 2017, pp. 1160–1166, http://dx.doi.org/10.1109/INDIN.2017.8104937.

[68] A. Theorin, K. Bengtsson, J. Provost, M. Lieder, C. Johnsson, T. Lundholm, B. Lennartson, An event-driven manufacturing information system architecture for industry 4.0, Int. J. Prod. Res. 55 (5) (2017) 1297–1311, http://dx.doi.org/10.1080/00207543.2016.1201604.

[69] G.E. Modoni, A. Trombetta, M. Veniero, M. Sacco, D. Mourtzis, An event-driven integrative framework enabling information notification among manufacturing resources, Int. J. Comp. Integ. M. 32 (3) (2019) 241–252, http://dx.doi.org/10.1080/0951192X.2019.1571232.

[70] P. Kannisto, D. Hästbacka, Asynchronous communication platform concept to coordinate large-scale industrial processes, IFAC-PapersOnline 51 (11) (2018) 1403–1408, http://dx.doi.org/10.1016/j.ifacol.2018.08.325, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, doi:10.1016/j.ifacol.2018.08.325.

[71] D. Hästbacka, P. Kannisto, M. Vilkko, Information models and information exchange in plant-wide monitoring and control of industrial processes, in: Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Vol. 3: KMIS, 2018, pp. 216–222, http://dx.doi.org/10.5220/0006960602160222.

[72] D. Hästbacka, P. Kannisto, M. Vilkko, Data-driven and event-driven integration architecture for plant-wide industrial process monitoring and control, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 2979–2985, doi:10.1109/IECON.2018.8591323.

[73] A. Burger, H. Koziolek, J. Rückert, M. Platenius-Mohr, G. Stomberg, Bottleneck identification and performance modeling of OPC UA communication models, in: ICPE '19, Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, Association for Computing Machinery, New York, NY, USA, 2019, pp. 231–242, http://dx.doi.org/10.1145/3297663.3309670.

[74] P.T. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2) (2003) 114–131, http://dx.doi.org/10.1145/857076.857078.

[75] J. O'Hara, Toward a commodity enterprise middleware, Queue 5 (4) (2007) 48–55, http://dx.doi.org/10.1145/1255421.1255424.

[76] RabbitMQ, Homepage, 2020, https://www.rabbitmq.com/ [Retrieved 23 Nov 2020].

[77] E. Trunzer, P. Prata, S. Vieira, B. Vogel-Heuser, Concept and evaluation of a technology-independent data collection architecture for industrial automation, in: IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, 2019, pp. 2830–2836, http://dx.doi.org/10.1109/IECON.2019.8927399.

[78] RFC 793, Transmission Control Protocol, Information Sciences Institute, University of Southern California, 1981, https://tools.ietf.org/html/rfc793 [Retrieved 27 May 2020].

[79] ITU-T X.805, Security Architecture for Systems Providing End-To-End Communications, International Telecommunication Union, 2003.

[80] B2MML, Business to Manufacturing Markup Language, Manufacturing Enterprise Solutions Association, 2013, http://www.mesa.org/en/B2MML.asp [Retrieved 22 Jun 2020].

[81] OGC 07-036r1, OpenGIS Geography Markup Language Encoding Standard (version 3.2.2 approved), Open Geospatial Consortium, 2016, http://www.opengeospatial.org/standards/gml [Retrieved 22 Jun 2020].

[82] OGC 08-094r1, SWE Common Data Model Encoding Standard (version 2.0.0 approved), Open Geospatial Consortium, 2011, http://www.opengeospatial.org/standards/swecommon [Retrieved 22 Jun 2020].

[83] OGC 10-004r3 and ISO 19156:2011(E), Geographic Information — Observations and Measurements (Version 2.0 Approved for Public Release), Open Geospatial Consortium, 2013, http://www.opengeospatial.org/standards/om [Retrieved 22 Jun 2020].

[84] OGC 15-043r3, Timeseries Profile of Observations and Measurements (Version 1.0 Approved for Public Release), Open Geospatial Consortium, 2016, http://www.opengeospatial.org/standards/tsml [Retrieved 22 Jun 2020].

[85] OGC 12-006, Sensor Observation Service Interface Standard (Version 2.0 Approved), Open Geospatial Consortium, 2012, http://www.opengeospatial.org/standards/sos [Retrieved 22 Jun 2020].

[86] OGC 09-000, Sensor Planning Service Implementation Standard (version 2.0 approved), Open Geospatial Consortium, 2011, http://www.opengeospatial.org/standards/sps [Retrieved 22 Jun 2020].

[87] P. Kannisto, A. Kätkytniemi, M. Vilkko, D. Hästbacka, Software toolkit for development of interoperable communications in data-driven systems, in: 17th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2021), 2021, pp. 844–849, in press.

[88] P. Kannisto, D. Hästbacka, K. Rainio, J. Leinonen, M. Alarotu, T. Pajula, J. Savolainen, M. Vilkko, Plant-wide communication architecture enabling online life cycle assessment, in: Automaatiopäivät 23, 2019, http://urn.fi/URN:NBN:fi:tuni-202102112034 [Retrieved 20 May 2021].

[89] H. Derhamy, J. Rönnholm, J. Delsing, J. Eliasson, J. van Deventer, Protocol interoperability of OPC UA in service oriented architectures, in: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), 2017, pp. 44–50, http://dx.doi.org/10.1109/INDIN.2017.8104744.

[90] Umsetzungsstrategie Industrie 4.0, Ergebnisbericht der Plattform Industrie 4.0, BITKOM e.V./VDMA e.V./ZVEI e.V., 2015, https://www.bitkom.org/Bitkom/Publikationen/Umsetzungsstrategie-Industrie-40.html [Retrieved 1 Jun 2020].